REMARKS

This communication is responsive to the Final Office Action dated November 18, 2009.

Claim Rejection Under 35 U.S.C. § 102

In the Final Office Action, the Examiner rejected Applicant's independent claims 1 and 24–26 under 35 U.S.C. 102 as being anticipated by four eparate prior art references. Specifically, the Examiner rejected claims 1 and 24-25 under 35 U.S.C. 102 as being anticipated by Reiche et al. (US 6,092,196), Aviani, Jr. et al. (US 6,532,493), Wein et al. (US 7,240,100) and Giles et al. (US 6,986,047). Applicant respectfully traverses the rejections. The applied references fail to disclose or suggest the inventions defined by Applicant's claims, and there would have been no apparent reason for modification to arrive at the claimed invention.

Prior to addressing the rejection, Applicant provides a brief summary of the prior art and the portions cited by the Examiner.

Reiche

Reiche describes authentication server for use in a data network. The portions of Reiche cited by the Examiner (cols. 5-6) describe a user using an "HTTP data exchange session" with a web browser to access a resource. [1] At this time the user is redirected to a central authentication server that determines whether the user is authorized to access the resource. Reiche states that the authentication server receives a response and compares data of the response to entries in a database of user IDs and passwords.[2] Reiche does not explain at all how this comparison is carried out.

Aviani

Aviani describes a network caching system.[3] When requested content is not in the cache, the cache attempts to establish a connection to the destination server to retrieve the content. At this time, the cache adds certain information to the HTTP header of the request. [4] The cited portions of Aviani (col. 6, ln. 20 – co. 8, ln. 26), describe the network cache modifying the HTTP

[1] Reiche, Summary.
[2] *Id.*
[3] Aviani, Summary.
[4] *Id.*

header so that the original client platform is identified for any subsequent caching systems encountered by the request. In relevant part, Aviani states that, upon reception of a new request by the same caching system, a client/server pair identified by the HTTP header is compared to a resident bypass list.[5] Aviani does not explain at all how this comparison is carried out.

Wein

Wein describes a content delivery network that serves content to customers. The cited portions of Wein (col. 5, ln. 10 – col. 6, ln. 59) state that the content delivery uses a header in an HTTP request to determine the appropriate metadata to locate and server.[6] The cited portions of Wein do no describe how the content delivery network processes the header and Applicant can find no relevant teachings with respect to Applicant's claims.

Giles

Giles describes methods for enabling access to a semi-trusted web server.[7] The cited portion of Giles (col. 5, ln. 12 – col. 6, ln. 58) describes correlating a client IP address and HTTP header information to a client credential. The cited portions of Giles do not describe how the correlation is performed and Applicant can find no relevant teachings with respect to Applicant's claims.

Applicant makes the following comments with respect to each of the 102 rejections set forth in the Office Action. For all of the rejections, the Examiner argues that the references inherently anticipate the elements of Applicant's independent claims 1 and 24-26 based on an assertion that: (1) each of the references make reference to processing HTTP communications, such as comparing information in an HTTP header, and (2) that processing of HTTP communications require "inherent string comparison operations" that anticipate Applicant's claims.[8]

[5] Aviani, col. 7, ll. 33-35.
[6] Wein, col. 6, ll. 54-60.
[7] Giles, Abstarct.
[8] See, e.g., Office Action, pg. 4 referring to HTTP string comparisons and stating "one of ordinary skilled [sic] in the art at the time of the invention very well understands that http strings are case insensitive and http string comparison contain [sic] inherent operations … ." See also Office Action, pg. 9, with respect to the rejection as anticipated by Aviani based inherency arguments; See, also, Office Action, pg. 13-14, with respect to the 102 rejection as anticipated by Wein based inherency arguments.

Applicant points out that the Examiner's general assertions that some network systems process HTTP communications, including comparing information to data in an HTTP header, do not provide substantial evidence that the particular techniques for performing the string comparisons described and claimed by the Applicant are unpatentable over such systems. The Examiner appears to be unwilling to consider the actual steps set forth in any of Applicant's claims with respect techniques for performing string comparisons in view of the teachings of the prior art. Instead, the Examiner appears to have concluded that any prior art network system that merely mentions processing HTTP strings must *per se* anticipate all other techniques for processing HTTP strings. This is clearly erroneous. In order to support an anticipation rejection under 35 U.S.C. 102, it is well established that a prior art reference must disclose each and every element of a claim. This well known rule of law is commonly referred to as the "all-elements rule."[9] If a prior art reference fails to disclose any element of a claim, then rejection under 35 U.S.C. 102(b) or (e) is improper.[10] Moreover, findings by the Examiner must be based upon substantial evidence, and not subjective musings or conjecture by the Examiner.[11]

Second, Applicant points out that the Examiner has incorrectly applied the law with respect to inherency. "Under principles of inherency, if the prior art necessarily functions in accordance with, or includes, the claimed limitation, it anticipates." In re Cruciferous Sprout Litigation, 64 USPQ2d 1202, 1206 (Fed. Cir. 2002) (emphasis added) (quoting MEHL/Biophile Int'l Corp. v. Milgraum, 192 F.3d 1362, 1365, 52 USPQ2d 1303, 1305 (Fed. Cir. 1999)). The Examiner erred in making his broad-sweeping assertion that HTTP string comparisons contain inherent operations and the cited prior art network system that process HTTP strings must, therefore, anticipate Applicant's claims. The Examiner has offered no substantial evidence to suggest that the prior art systems necessarily perform HTTP string comparisons in the manner set forth in Applicant's claims.

More specifically, with respect to each of the rejections under 35 U.S.C. 102, the Examiner relies on the following to show the inherent operations performed by the cited prior art: (1) Knowledge Base Document String Compare Function ("the Knowledge reference"), and

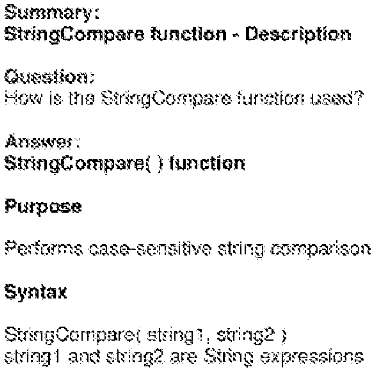
[9] *See Hybritech Inc. v. Monoclonal Antibodies, Inc.*, 802 F.2d 1367, 231 USPQ 81 (CAFC 1986) ("it is axiomatic that for prior art to anticipate under 102 it has to meet every element of the claimed invention").

[10] *Id. See also Lewmar Marine, Inc. v. Barient, Inc.* 827 F.2d 744, 3 USPQ2d 1766 (CAFC 1987); *In re Bond*, 910 F.2d 831, 15 USPQ2d 1566 (CAFC 1990); *C.R. Bard, Inc. v. MP Systems, Inc.*, 157 F.3d 1340, 48 USPQ2d 1225 (CAFC 1998); *Oney v. Ratliff*, 182 F.3d 893, 51 USPQ2d 1697 (CAFC 1999); *Apple Computer, Inc. v. Articulate Systems, Inc.*, 234 F.3d 14, 57 USPQ2d 1057 (CAFC 2000).

[11] *Id.*

(2) www.cyber.com for inherent operations when such systems perform HTTP string comparisons ("the Cyber reference").

The Knowledge reference is a two page document that merely summarizes the parameters passed to, and the results from, a string compare software function without any explanation as to the comparison that is implemented. A portion of the Knowledge reference is reproduced below:



Summary:
StringCompare function - Description

Question:
How is the StringCompare function used?

Answer:
StringCompare( ) function

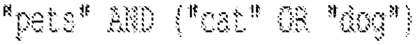
Purpose

Performs case-sensitive string comparisons.

Syntax

StringCompare(string1, string2)
string1 and string2 are String expressions

The Cyber reference is a whitepaper by Cybersoft that describes the "CyberSoft Virus Description Language." According to CyberSoft, language allows a user to express a definition for a virus or pattern.[12] The Cyber reference provides an example of how a user could express such a virus definition using the language:



```
"pets" AND ("cat" OR "dog")
```

The virus scanner and pattern analysis tool sold by CyberSoft processes these virus definitions to look for attacks in network communications.[13] Notably, the Cyber reference describes only the language in which virus definition can be expressed by a user. The Cyber reference does not describe the techniques by which the virus scanner and pattern analysis tool actually compare strings. For example, with respect to the example expression above of "pets" and ("cat" or "dog"), the Cyber reference states that a file matches this virus expression if the file contains the word "pets" anywhere in the file and the file also contains the word "cat" or the word "dog" anywhere in the file. This virus description language provides no teaching as to what operations the system performs, or in what order, to actually implement the analysis.

The summary of a string comparison function in Knowledge and the virus description language of Cyber certainly provide no substantial evidence that the systems of Reiche, Aviani,

[12] http://www.cyber.com/whitepapers/papers/cvdl.shtml
[13] http://www.cyber.com/whitepapers/papers/cvdl.shtml

Wein or Giles necessarily include the elements set forth in Applicant's claims. For example, Applicant's claim 1 requires performing a bitwise exclusive OR operation to the two strings for which a case-insensitive match is being generated, i.e., between an ASCII binary representation of at least a segment of the unknown string and an ASCII binary representation of at least a segment of the predefined string. In other words, the XOR operation of claim 1 is applied to the strings being compared, and those strings being compared are the inputs to that XOR operation. Claim 1, as a whole, then requires performing a bitwise operation on a predefined flag and a result of that XOR operation, and comparing the predetermined flag and a result of the bitwise OR operation to produce an indication for the case-insensitive string match. The Examiner has provided no substantial evidence that suggests that any of the Reiche, Aviani, Wein, and Giles references necessarily compare a known string and an unknown string in the specific manner set forth in claim 1. As an example, Applicant cannot find any teaching in the references that involve the use of a predefined string, let alone the process set forth in claim 1.

To further aid the Examiner understanding of the claimed invention, Applicant submits that there may be many algorithms for performing a case-insensitive comparison of a known string to an unknown string. Many algorithms may rely on first converting both strings to a common case, such as converting both to upper case or converting both to lower case. After converting the strings, the individual bytes of each string can then be compared to test for equality. These techniques, however, are costly in terms of memory cycles and CPU operations in that they require byte-wise conversion of each character in the string to a common case.[14] This conversion process may be performed by performing a comparison and then an addition or subtraction operation on each character. Applicant's claimed technique may avoid the disadvantages of these time-consuming approaches for converting the strings to a common case. The Examiner's analysis with respect to the rejections under 35 U.S.C. § 102 effectively read out all of the elements of Applicant's claims by asserting that the entire algorithm set forth in claim 1, for example, is inherent in the prior art.

For at least these reasons, the Examiner has failed to establish a prima facie case for anticipation of Applicant's claims under 35 U.S.C. 102. Withdrawal of this rejection is requested.

[14] For examples and discussions, see http://stackoverflow.com/questions/11635/case-insensitive-string-comparison-in-c, see also http://lafstern.org/matt/col2_new.pdf.

Claim Rejection Under 35 U.S.C. § 103

Branstad (US 6,842,860) in view of Fielding, www.cyber.com, pages 3-12, August 11, 2001, and "Official Notice"

In the Final Office Action, the Examiner rejected claims 1–5, 7, 8, 11–20, 22 and 23 under 35 U.S.C. 103(a) as being unpatentable over Branstad (US 6,842,860) in view of Fielding, www.cyber.com, pages 3-12, August 11, 2001, and "Official Notice."

With respect to these elements, the Examiner asserted that Branstad at col. 21, ll. 3-27 and col. 3, ll. 26-38 teaches storing, on a network device, a database containing a plurality of predefined strings, wherein the predefined strings stored within the database represent known headers for a network communication protocol. *Final Office Action, pg. 23.* Branstad does not describe methods and devices for matching strings. Instead, Branstad describes an authentication system that authenticates messages exchanged between devices. *Branstad, Summary.* Branstad at col. 3, ll. 26-38 describes a network device that "generates" an authentication tag 140 for an outbound packet that is being sent by the Branstad device. Branstad at col. 21, ll. 3-27, also recited by the Examiner, appears to describe functions supported by the Branstad device for verification of the message by the receiver.

Applicant's claim 1 recites, in part, a method that requires storing, on a network device, a plurality of predefined strings within a database, and in response to receiving the network message, selecting one of the plurality of predefined strings stored within the database of the network device. Thus, the predefined string recited in Applicant's claim 1 must be selected from a database that stores a plurality of predefined strings and, therefore, cannot be a portion of the network message received by the device, as suggested by the Examiner on page 3 of the Office Action. Rather, Applicant's claim 1 makes clear that one of the pre-defined strings is selected from the database in response to that device receiving a network message.

To be clear, Branstad's message verification technique cited at col. 3, ll. 26-38 by the Examiner relies on an authentication tag that is generated based on the contents of a network packet. This does not teach or suggest storing, on a network device, a database containing a plurality of predefined strings, wherein the predefined strings stored within the database represent known headers for a network communication protocol, as required by claim 1.

Moreover, this does not teach or suggest in response to receiving the network message, selecting one of the plurality of predefined strings stored within the database of the network device, as required by claim 1. Branstad provides no teaching of such features. Rather, the Branstad authentication tag described by the portions of Branstad cited by the Examiner is dynamically computed based on the particular contents of the packet being sent.

Further, Applicant's claim 1 requires that the predefined strings stored within the database represent "known headers for a network communication protocol." Again, the Examiner relies on Branstad for such teachings, referring to col. 3 at col. 21, ll. 3-27 and col. 3, ll. 26-38. However, these portions of Branstad make clear that the authentication tag (which is compared to a different authentication tag in Branstad) is generated (i.e., computed) from the contents of a network packet using cryptographic techniques.[15] In no manner can the cryptographic authentication tags generated by Branstad based on the content of packets be considered predefined strings that represent known headers for a network communication protocol, as required by claim 1. To the contrary, even in view of the other cited references, any string generated by a cryptographic process simply cannot be a known header for a communication protocol.

Applicant's claim 1 further requires identifying a portion of the network message as an unknown string for comparison with the selected predefined string. Thus, the literal language of claim 1 limits this claimed embodiment to a method where the predefined string is selected from a database of predefined strings in response to the received message, i.e., after a message is received, and that the unknown string is a portion of that received network message. These features are not taught or suggested by the combination of references cited by the Examiner. Specifically, with respect to these elements the Examiner again refers to Branstad col. 3, ll. 26-38 that, as stated above, describes the generation of an authentication tag for inclusion within an outbound packet being produced by the sending device. The receiving device in the Branstad system then extracts this authentication tag upon receiving the packet.

Furthermore, Applicant's method of claim 1 for comparing an unknown string to a predefined string further requires performing a bitwise exclusive OR operation between an ASCII binary representation of at least a segment of the unknown string and an ASCII binary representation of at least a segment of the selected predefined string. In this way, claim 1

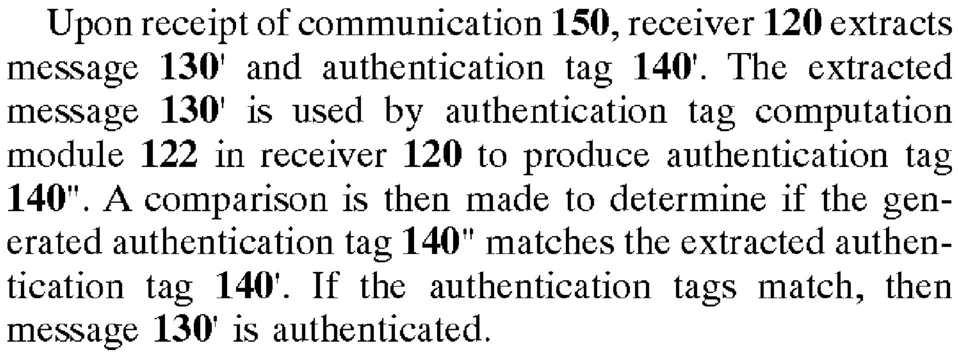
[15] Branstad at col. 21, ll. 4-46,

literally requires that a bitwise exclusive OR operation be performed between: (1) a segment of the unknown string that was identified within a network message, and (2) a segment of the predefined string that was selected from a database of predefined strings in response to the message. Claim 1 also requires the additional step of performing a bitwise operation between a predefined flag and a result of the exclusive OR operation. The Examiner cites Branstad at cols. 19-22 with respect to the step of performing a bitwise exclusive OR operation between an ASCII binary representation of at least a segment of the unknown string and an ASCII binary representation of at least a segment of the selected predefined string. However, in these columns Branstad describes cryptographic techniques by which the sender generates an authentication tag for an outbound packet. See Branstad, cols. 19-22. Although this process may involve an XOR operation, the cryptographic techniques for generating an authentication tag from a packet fails to teach the requirements of claim 1 of performing a bitwise exclusive OR operation between an ASCII binary representation of at least a segment of the unknown string and an ASCII binary representation of at least a segment of the selected predefined string.

In addition, Applicant's claim 1 is limited to performing a bitwise exclusive OR operation to the two strings for which a case-insensitive match is being generated, i.e., between an ASCII binary representation of at least a segment of the unknown string and an ASCII binary representation of at least a segment of the predefined string. In other words, the XOR operation of claim 1 is applied to the strings being compared, and those strings being compared are the inputs to that XOR operation. Claim 1, as a whole, requires performing a bitwise operation on a predefined flag and a result of the XOR operation, and comparing the predetermined flag and a result of the bitwise OR operation to produce an indication for the case-insensitive string match. The combination of references cited by the Examiner fails to teach or suggest this process for numerous reasons.

Specifically, none of the references, either singularly or in combination, describe any method that determines whether an unknown string matches a predefined string by performing an XOR operation between the predefined string and the unknown string. As stated in Applicant's previous response, the Examiner's reasoning is logically flawed in asserting that Branstad teaches performing a bitwise exclusive OR operation between at least a segment of the predefined string and a segment of the unknown string when comparing those two strings. Branstad teaches application of an XOR operation to portions of a message so as to **compute** a binary string, i.e., cryptographic authentication tag. Branstad makes abundantly clear that the XOR operation is not used as part of a comparison operation of the two strings that are provided as inputs to the XOR operation, as required by claim 1. Rather, Branstad describes application of XOR operations only for computing an authentication tag. Only after the tag is generated using cryptographic techniques (which may involve an XOR) is it then compared in a conventional way to a different string, i.e., the authentication tag extracted from the received message.

Col. 3, ll. 35-43 Branstad makes clear that a single authentication tag is computed from an inbound message (by use of cryptographic techniques that may use an XOR as Branstad explains). It is this process that is described in Branstad at col. 19-22 for computing an authentication tag from a packet. Branstad describes a separate comparison operation performed to compare that computed authentication tag to a different authentication tag extracted from the same message:



> Upon receipt of communication **150**, receiver **120** extracts message **130'** and authentication tag **140'**. The extracted message **130'** is used by authentication tag computation module **122** in receiver **120** to produce authentication tag **140"**. A comparison is then made to determine if the generated authentication tag **140"** matches the extracted authentication tag **140'**. If the authentication tags match, then message **130'** is authenticated.

Thus, in no manner does Branstad teach or suggest performing a bitwise XOR operation between a predefined string and an unknown string so that the result of that XOR operation could be used as an indication of a match between those two strings that were applied as inputs to that XOR operation, as required by claim 1.

In other words, the Examiner's argument is based on the premise that different portions of the received message in Branstad could be viewed as two strings, and that Branstad suggests applying an XOR operation to those two strings when computing the cryptographic authentication tag. Even assuming this is a valid interpretation of Branstad, in no way does Branstad in view of the other cited references utilize an XOR operation in a process where the result of the XOR operation could be used to provide an indication that a case-insensitive match exists between those two input strings or the XOR operation, as required by claim 1. For these reasons, Branstad fails to teach or suggest performing a bitwise exclusive OR operation between an ASCII binary representation of at least a segment of the unknown string and an ASCII binary representation of at least a segment of the selected predefined string, performing a bitwise operation between a predefined flag and a result of the exclusive OR operation, and comparing the predefined flag and a result of the bitwise operation to produce an indication for a case-insensitive string match, wherein the indication for the case-insensitive string match indicates whether all characters of the unknown string within the network message match all corresponding characters of the selected predefined string so as to match one of the known headers of the network communication protocol.

The combination of references cited by the Examiner fails to address this flaw in the Examiner's reasoning with regard to Branstad. Fielding merely describes the HTTP protocol and, in particular, the parameters and headers used when communicating via the HTTP protocol. Fielding does nothing to overcome the fundamental deficiencies of Branstad relative to Applicant's claim 1. Moreover, the Cyber reference, as discussed above, describes only the language in which virus definition can be expressed by a user and does not describe any techniques by which the virus scanner and pattern analysis tool actually compare strings.

Modification of the Branstad authentication technique in view of Fielding, as suggested by the Examiner, would result only in an authentication system in which the messages sent over the network conform to the HTTP protocol. The cryptographic technique used in Branstad to generate the authentication tag from an HTTP message would be the same. To the extent the cryptographic technique (e.g., KR5) of Branstad uses XOR operations and rotations, the operations would be applied to the HTTP message to compute the authentication tag in the system proposed by the Examiner. Fielding provides no suggestion to use an XOR operation for any other purpose, and the combination of Branstad in view of Fielding fails to teach or suggest applying a bitwise XOR operation between two different messages when determining whether the strings match.

Further, the Examiner rejects claim 1 based on significant "Official Notice":

> *"Official Notice" is taken that both the concept and advantages of providing usage of the string being predefined, performing a bitwise operation between the predefined flag and the result and comparing the predefined flag and a result of the*

> *bitwise operation, wherein the indication for the case-insensitive string match indicates whether all characters of the unknown string within the network message match all corresponding characters of the selected predefined string so as to match one of the known headers, processing the network message based on the indication of the match and outputting a response from the network device based on the processed network message is well known and expected in the art.*[16]

Applicant traverses this Official Notice. Based on the Examiner's statements, Saccoccio, cited by the Examiner for support of this Official Notice, appears only to describe a need for case insensitive comparisons without describing any particular solution, let alone Applicant's claimed technique.

Reiche-Nortel-Networks in view of Branstad (US 6,842,860) Fielding, www.cyber.com, pages 3-12, August 11, 2001, and "Official Notice"

The Examiner rejected claims 2–5, 7, 8, 11–20, 22 and 23 under 35 U.S.C. 103(a) as being unpatentable over Reiche in view of Branstad, Fielding, www.cyber.com, pages 3-12, August 11, 2001, and "Official Notice." As discussed above, Reiche describes an authentication server that receives a response and compares data of the response to entries in a database of user Ids and passwords.[17] Reiche does not explain at all how this comparison is carried out. Thus, Reichi fails to overcome any of the deficiencies set forth above with respect to the independent claims.

For at least these reasons, the Examiner has failed to establish a prima facie case for non-patentability of Applicant's claims under 35 U.S.C. 103(a). Withdrawal of this rejection is requested.

[16] Office Action, pg. 25.

[17] *Id.*

CONCLUSION

All claims in this application are in condition for allowance. Applicant respectfully requests reconsideration and prompt allowance of all pending claims. Please charge any additional fees or credit any overpayment to deposit account number 50-1778. The Examiner is invited to telephone the below-signed attorney to discuss this application.

Date: ___February 18, 2010____	By: _____/Kent J. Sieffert/__________
SHUMAKER & SIEFFERT, P.A.	Name: Kent J. Sieffert, Reg. No.: 41,312

SHUMAKER & SIEFFERT, P.A.
1625 Radio Drive, Suite 300
Woodbury, Minnesota 55125
Telephone: 651.286.8341
Facsimile: 651.735.1102